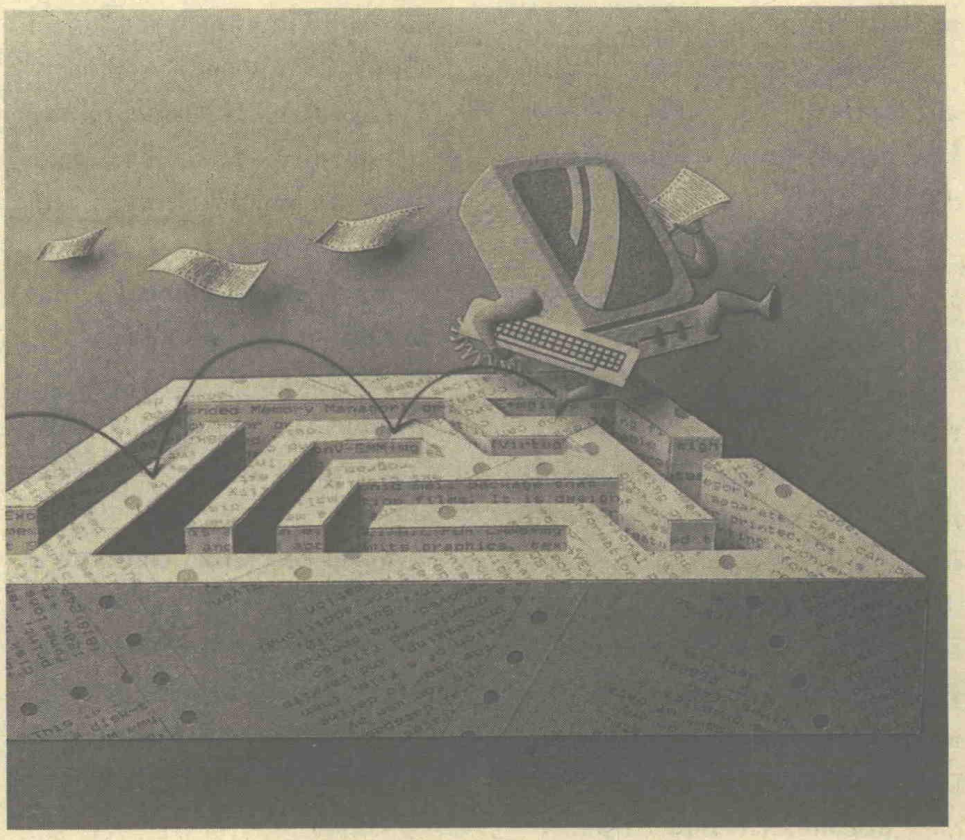
ศึกษาการวางแผนและการออกแบบโปรแกรม ตอนที่ 1

ออกแบบโปรแกรมด้วยการวางแผน การออกเอกสาร เพื่อแก้ปัญหาที่ยุ่งยาก



ถึงเวลาแล้วที่เราควรจะเริ่มการวางแผน ออกเอกสาร และทดสอบ ในรูปที่ 1 เป็นขั้นตอนการพัฒนาโปรแกรม

การออกแบบและการวางแผนการเขียนโปรแกรม

เราจะลองถกกันถึงการวางแผนและการทำเอกสารกันก่อน คำว่าเอกสาร (documentation) คือ ปัญหาและกระบวนการแก้ปัญหา ซึ่งจะบันทึกเก็บไว้ทั้งโปรแกรมเมอร์และไม่ใช่โปรแกรมเมอร์

โดยทั่วไปเราจะหมายถึงการวางแผนโปรแกรม การออกเอกสารตามแผน และการลงมือเขียนโปรแกรม ถ้าโปรแกรมที่เขียนนั้นเป็นการแก้ปัญหาที่ไม่ยากนักก็ไม่จำเป็นต้องวางแผน แต่ผมแนะนำให้วางแผนทุกครั้งก่อนลงมือเขียนโปรแกรมด้วยเหตุและผลหลายประการดังนี้

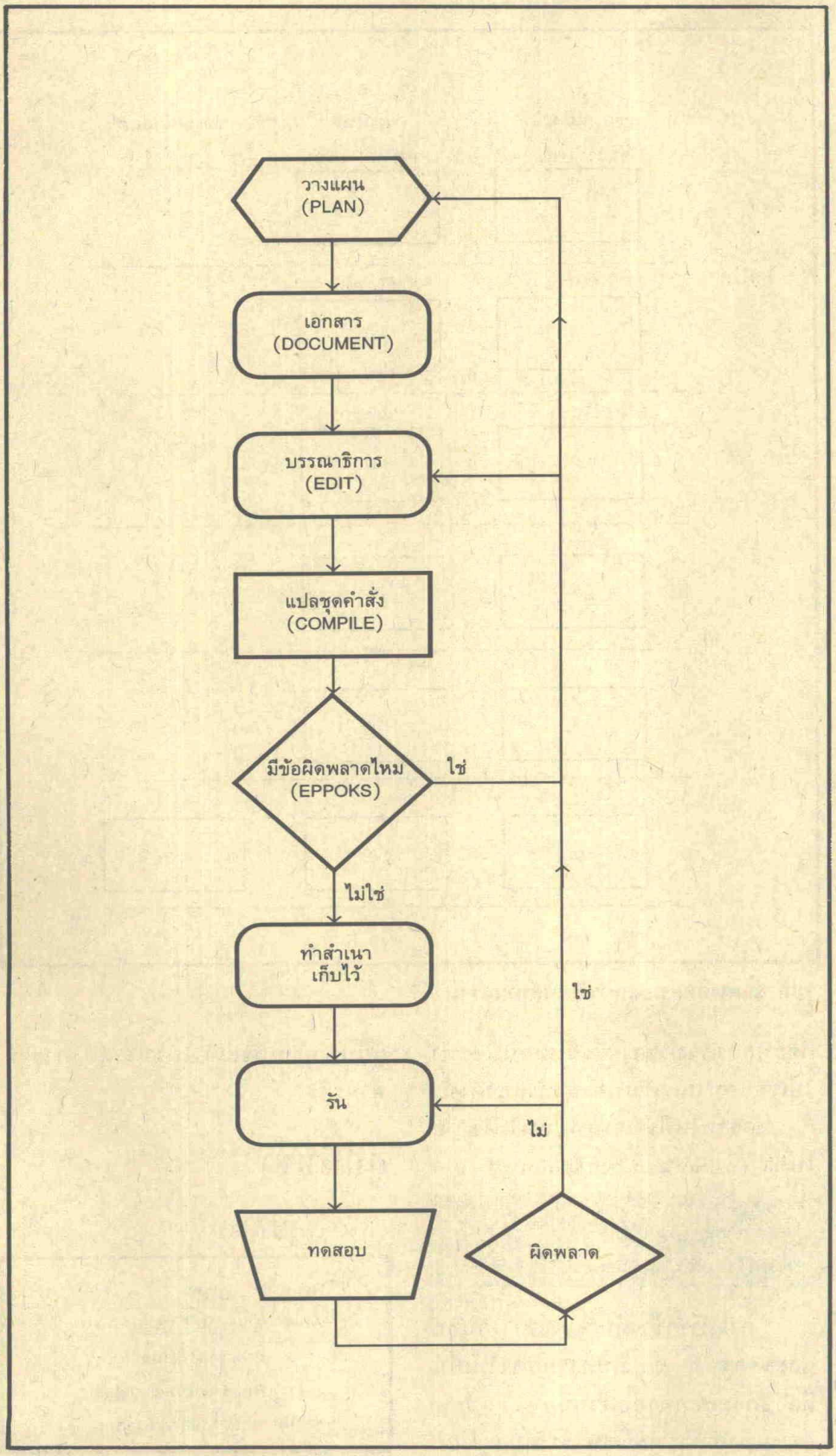
1. การวางแผนการออกเอกสารใช้สำหรับโปรแกรมใหญ่ ๆ ได้ดี

2. โปรแกรมเล็ก ๆ ก็จำเป็นที่จะต้องวางแผนให้ดีก่อน เพื่อภายหลังโปรแกรมเล็ก ๆ ต้องการจะขยายเป็นโปรแกรมใหญ่ หรือต้องการจะแก้ไขก็จะทำได้ง่ายขึ้น

ลองพิจารณาการหาผลบวกของเลขชุดหนึ่งที่เป็นจำนวนเต็มบวก การสร้างข้อความเอกสารที่ดีก็ทำตอนเขียนโปรแกรมที่ใช้เอดิเตอร์จากเทอร์โบ โดยใช้ชื่อไฟล์ชื่อเดียวกับชื่อของไฟล์โปรแกรม เช่น เราตั้งชื่อโปรแกรมว่า ADDNUM.PAS เราก็สร้างไฟล์เอกสารชื่อ ADDNUM.DOC

ก่อนจะพิมพ์ข้อความปัญหา ควรจะต้องรวมเอาข่าวสารเกี่ยวกับชื่อโปรแกรมและไฟล์โปรแกรม

```
DOCUMENTED PROGRAM :
        add_numbers
FILE NAME :
        ADDNUM.PAS
```



รูปที่ 1 แสดงขั้นตอนการพัฒนาโปรแกรม

ข้อมูลนำเข้าและข้อมูลนำออก (Input and Output)

ขั้นต่อมาเราก็ตัดสินว่าข้อมูลอะไรที่ต้องการจะนำเข้าสู่คอมพิวเตอร์ และข้อมูลอะไรที่ต้องการนำออกหน้าจอ ซึ่งเราต้องคิดไว้ล่วงหน้าก่อนว่าต้องการให้ผู้ใช้ป้อนตัวเลขที่จะให้บวกกี่จำนวน หรือจะใช้สัญญาณอะไรเป็นตัวบอกให้เลิกป้อนข้อมูล จะใช้ผู้ป้อนบรรทัดละจำนวนหรือหลาย ๆ จำนวนต่อบรรทัด

INPUT :

ชุดของจำนวนเต็มบวก โดยป้อนตัวเลขจำนวนละ 1 บรรทัด ถ้าต้องการจะเลิกป้อนให้พิมพ์เลข 0

OUTPUT :

ผลบวกของเลขทั้งหมดที่ป้อนเข้ามา ในรูปแบบดังนี้

ผลบวกของมันคือเลขจำนวนเต็ม

กระบวนการ

เราทราบแล้วว่าจะทำอะไร ขั้นต่อมาก็คือจะทำอย่างไรกับมันดี ซึ่งส่วนนี้จะเป็นส่วนที่ยาก จึงต้องเขียนเป็นกระบวนการไว้เตือนความจำของเราก่อน คือ

พิมพ์คำชี้แจงแก่ผู้ใช้

ป้อนข้อมูลทีละจำนวนต่อ 1 บรรทัด

สำหรับข้อมูลแต่ละบรรทัดไม่เท่ากับ 0 ให้รวมกันแล้วเก็บไว้ในที่ใดที่หนึ่ง ถ้าพบว่ามีบรรทัดหนึ่งเป็น 0 ให้พิมพ์ผลบวกทั้งหมดออก

ลองเขียนอัลกอริทึมแบบลำลอง

การเขียนข่าวสารภายนอกโปรแกรมจะเป็นการบ่งบอกถึงวิธีการแก้

ปัญหา โดยการเขียนนั้นไม่จำเป็นต้องถูกไวยากรณ์ทางภาษา และควรให้กะทัดรัดและปฏิบัติตามได้ง่าย การจะคำนวณและเก็บผลลัพธ์ไว้ที่ใดอย่าลืมกำหนดค่าเริ่มต้นก่อนเสมอ เช่น

กำหนดค่าของ total ด้วยค่า 0 แล้วลองทำด้วยมือ (คำนวณโดยไม่ใช้คอมพิวเตอร์) ดังรูปที่ 2

จากรูป 2 (a) จะเห็นว่าในช่องสี่เหลี่ยมว่างเปล่า

จากรูป 2 (b) เริ่มใส่ค่า 0 ลงใน total

ซึ่งจะเขียนคำสั่งลำลองดังนี้

```
print "Type any number of positive integers, one to a line"
```

พิมพ์ข้อความบนจอเพื่อให้ผู้ใช้ป้อนจำนวนเต็มบวกบรรทัดละจำนวน

```
print "When you are done, type 0"
```

เมื่อต้องการเลิกป้อนให้ใส่เลข 0

```
Read the first number
```

อ่านจำนวนแรกเข้ามา

ก็จะเข้ามาถึงส่วนที่เป็น 2 (c) คือการบวกค่าที่ป้อนเข้ามากับ 0 แล้วเก็บไว้ที่ total

ดังนั้นจากรูปส่วนที่เป็น 2 (d) จะเขียนคำสั่งลำลองได้ดังนี้

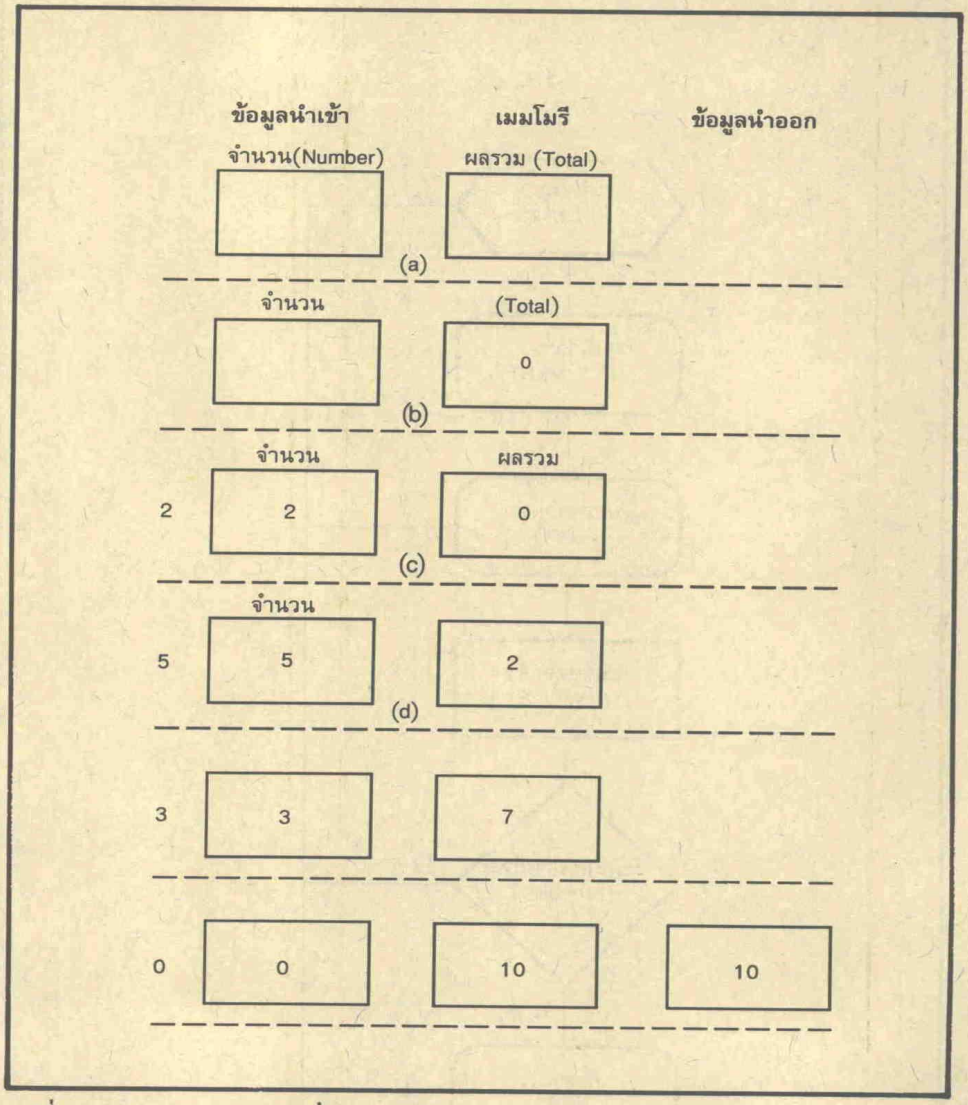
บวกจำนวนที่ป้อนเข้ามา แล้วเก็บไว้ที่ total

```
Add number to total
```

อ่านตัวเลขตัวอื่นเข้ามา

```
Read another number
```

พอถึงขั้นตอนจากรูป 2 (e) เราก็ต้องซ้ำขั้นตอน 2 ขั้นตอนข้างต้น แล้วทำซ้ำจนถึงขั้นตอน 2 (f) คือถ้าป้อน 0



รูปที่ 2 การติดตามผลการรันเพื่อหาผลรวม

ก็จะหยุดการทำซ้ำ ซึ่งขั้นตอนนี้จะทำไปเรื่อย ๆ และเขียนคำสั่งลำลองดังนี้

ตราบใดที่จำนวนที่ป้อนไม่ใช่ 0 ให้ทำ (while number is not 0)

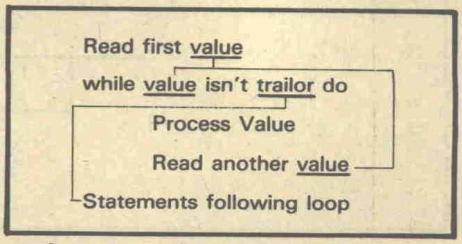
```
Add number to total
Read another number
```

การทำซ้ำข้างต้นเรียกว่าการวนลูป และจากรูปที่ 2 เป็นการแสดงให้เห็นถึงวิธีการสะกดรอยด้วยมือ

ซึ่งสิ่งที่เรากล่าวมาข้างต้นจะเขียนไฟล์เก็บไว้เพื่อทำให้เราจำได้ และผู้อื่นมาอ่านก็จะเข้าใจว่าเราต้องการจะทำอะไร

ลูปนำเข้า

(ดูรูปที่ 3)



รูปที่ 3 รูปแบบทั่วไปของลูปนำเข้า

```
program add_numbers;
{ find the sum of a list of positive integers}
{
 INPUT:
   A list of positive integers,entered one to a line. The
   end of the list is signaled by typing the number 0.
  OUTPUT:
   The sum of the numbers,in the format below:
      The sum of the numbers is ttttt
 }
 var
   number : integer;
   total  : integer;
 begin
  {initialize}
   total:=0;
  {Print instruction}
   writeln('Type any number of positive integers one to a line');
   writeln('When you are done,type 0);
  {Obtain numbers,keeping running total}
  readln(number);
  while number<> 0 do
     begin
      total :=total+number;
      readln(number)
     end;
   {Print result}
  writeln('The sum of the numbers is',total:5);
 end.
```

รูปที่ 4 โปรแกรม add_numbers

โปรแกรมเอกสาร :

add_numbers

ชื่อไฟล์ :

ADDNUM.PAS

ประโยคปัญหา :

หาผลบวกของจำนวนเต็มบวกชุดหนึ่ง

ข้อมูลนำเข้า :

ชุดของจำนวนเต็มบวกที่ป้อนเข้าทางคีย์บอร์ด 1 จำนวนต่อ 1 บรรทัด เมื่อต้องการเลิกป้อนให้พิมพ์ตัวเลข 0

ข้อมูลนำออก :

ผลบวกของจำนวนทั้งหมดที่ป้อนเข้ามา

กระบวนการ :

พิมพ์คำชี้แจงให้แก่ผู้ใช้ รับจำนวนเข้ามาทีละจำนวนที่ไม่ใช่จำนวนลบแล้วบวกเข้าด้วยกัน ถ้าอ่านพบจำนวนนั้นเป็น 0 ให้พิมพ์ผลลัพธ์ที่ได้จากการบวกออกมา

คำสั่งลำลอง :

1. ให้ total มีค่าเป็น 0
2. พิมพ์ "Type any number of positive integers one to a line"
3. พิมพ์ "When you are done, type 0"
4. อ่านข้อมูล (จำนวนเต็มบวก) ตัวแรก
5. ตราบใดที่จำนวนที่อ่านเข้ามาไม่เป็น 0 ให้ทำดังนี้
   5.1 บวกจำนวนนั้นกับ total
   5.2 อ่านจำนวนอื่นเข้ามา
6. พิมพ์ผลลัพธ์

หลังจากนั้นก็นำเอาคำสั่งลำลองมาเขียนโปรแกรมทางคอมพิวเตอร์ อาจจะใช้ภาษาใดก็ได้ เช่น เบสิก ปาสคาลหรือซี แต่ในบทความนี้จะเขียนด้วยปาสคาล เพราะต้องการปูพื้นเกี่ยวกับภาษาโครงสร้างปาสคาลด้วย และคอมพายเลอร์ที่ใช้ก็เป็นเทอร์โบ-ปาสคาล